Московский Авиационный Институт

(Национальный Исследовательский Университет)

Институт информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

**Лабораторная работа №3 по курсу**

**«Компьютерная графика»**

Выполнил: Лисин Р.С.

Группа: М8О-306Б-20

Преподаватель: Филиппов Г.С.

Дата: \_\_\_\_\_\_\_\_\_\_\_

Оценка: \_\_\_\_\_\_\_\_\_\_\_

Подпись: \_\_\_\_\_\_\_\_\_\_\_

Москва, 2022

**Лабораторная работа №3**

**Основы построения фотореалистичных изображений.**

**Задание**: Используя результаты Л.Р.№2, аппроксимировать заданное тело выпуклым многогранником. Точность аппроксимации задается пользователем. Обеспечить возможность вращения и масштабирования многогранника и удаление невидимых линий и поверхностей. Реализовать простую модель закраски для случая одного источника света. Параметры освещения и отражающие свойства материала задаются пользователем в диалоговом режиме.

**Вариант 5:** Эллипсоид

**Описание**

Для выполнения данного задания я использовал библиотеку matplotlib для Python. Эта библиотека позволяет довольно просто работать с трехмерными фигурами. Также matplotlib имеет автоматическое масштабирование.

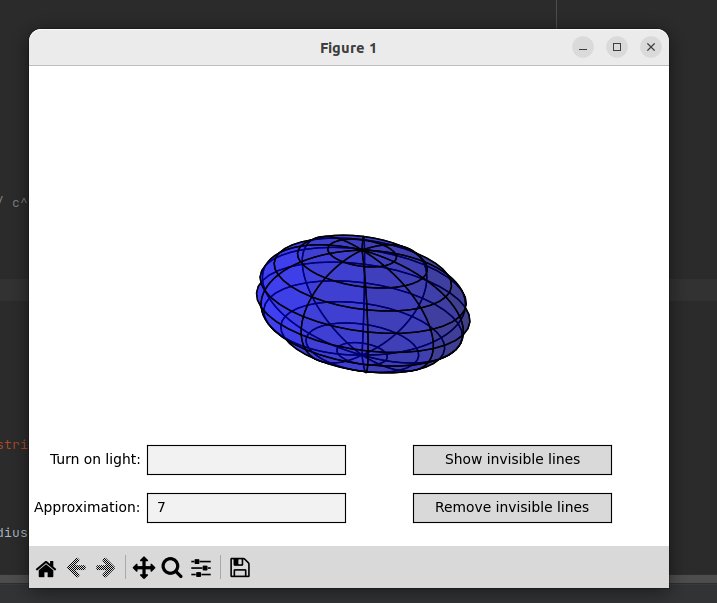
Для отрисовки эллипсоида я перешёл в сферические координаты. Аппроксимацию я сделал через пропуск точек при отрисовке фигуры с определённым интервалом (rstride, cstride). Для удаления невидимых линий я просто меняю заливку фигуры. В качестве освещения эллипсоида я использую класс LightSource модуля matplotlib с оттенками в цвете меди.

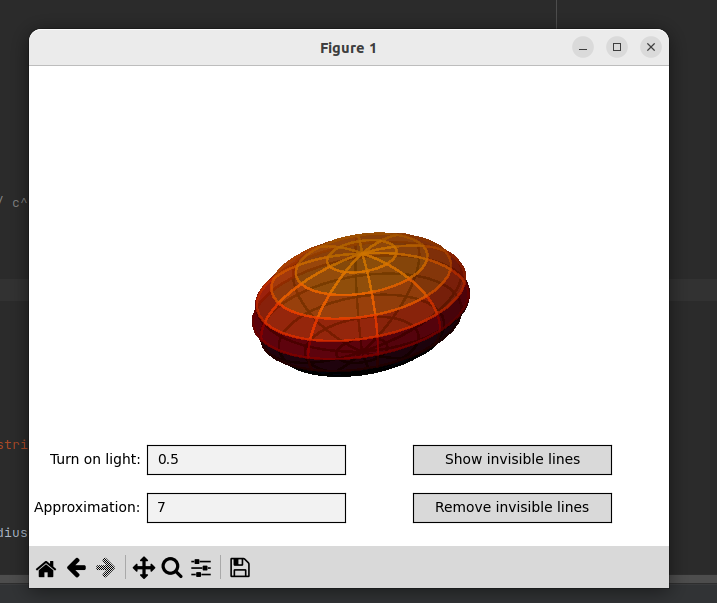
Для обработки пользовательского ввода я использовал виджеты matplotlib Button и TextBox, которые позволяют легко добавить кнопки и текстовый ввод в окно соответственно.

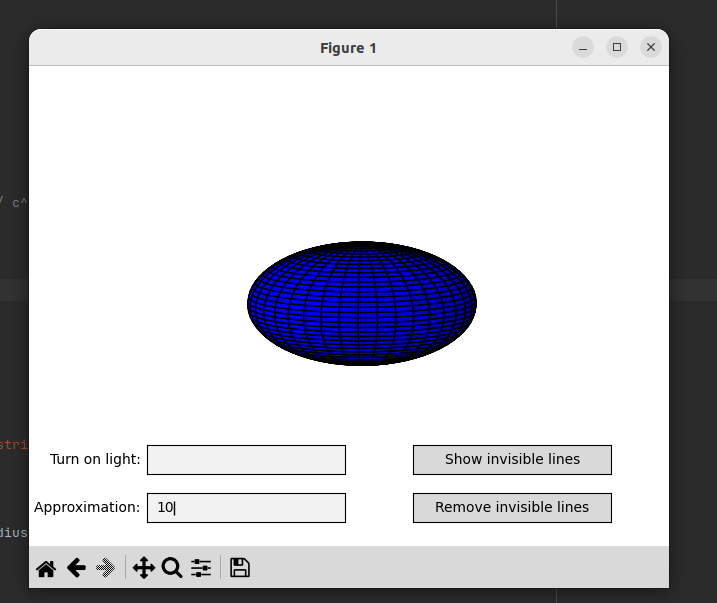
**Исходный код**

import matplotlib.pyplot as plt  
import numpy as np  
from matplotlib.widgets import Button, TextBox  
from matplotlib.colors import LightSource  
  
approximation = 4  
alpha = 0.5  
steps = 11  
fig = plt.figure()  
ax = fig.add\_subplot(111, projection='3d')  
  
# Ellipsoid equation: x^2 / a^2 + y^2 / b^2 + z^2 / c^2 = 1  
coefficients = (1, 2, 2)  
rx, ry, rz = 1 / np.sqrt(coefficients)  
  
u = np.linspace(0, 2 \* np.pi, 40)  
v = np.linspace(0, np.pi, 40)  
  
x = rx \* np.outer(np.cos(u), np.sin(v))  
y = ry \* np.outer(np.sin(u), np.sin(v))  
z = rz \* np.outer(np.ones\_like(u), np.cos(v))  
  
ax.plot\_surface(x, y, z, rstride=approximation, cstride=approximation, color='b', alpha=alpha, edgecolors="black")  
  
max\_radius = max(rx, ry, rz)  
for axis in 'xyz':  
 getattr(ax, 'set\_{}lim'.format(axis))((-max\_radius, max\_radius))  
  
  
def button\_callback\_remove(event):  
 global alpha  
 alpha = 1  
 ax.plot\_surface(x, y, z, rstride=approximation, cstride=approximation, color='b', alpha=alpha, edgecolors="black")  
 plt.draw()  
  
  
button\_ax\_remove = fig.add\_axes([0.6, 0.05, 0.31, 0.06])  
button\_remove = Button(button\_ax\_remove, "Remove invisible lines")  
button\_remove.on\_clicked(button\_callback\_remove)  
  
  
def button\_callback\_show(event):  
 global alpha  
 alpha = 0.5  
 ax.plot\_surface(x, y, z, rstride=approximation, cstride=approximation, color='b', alpha=alpha, edgecolors="black")  
 plt.draw()  
  
  
button\_ax\_show = fig.add\_axes([0.6, 0.15, 0.31, 0.06])  
button\_show = Button(button\_ax\_show, "Show invisible lines")  
button\_show.on\_clicked(button\_callback\_show)  
  
  
def turn\_on\_light(intensitivity=1):  
 ls = LightSource()  
 illuminated\_surface = ls.shade(z, plt.cm.copper, fraction=float(intensitivity))  
 ax.plot\_surface(x, y, z, rstride=approximation, cstride=approximation, color='b', alpha=alpha, edgecolors="black", antialiased=False, facecolors=illuminated\_surface)  
 plt.draw()  
  
  
light\_box = fig.add\_axes([0.184, 0.15, 0.31, 0.06])  
text\_box\_light = TextBox(light\_box, "Turn on light: ")  
text\_box\_light.on\_submit(turn\_on\_light)  
  
  
def change\_approximation(new\_approximation):  
 global approximation, alpha  
 approximation = steps - int(new\_approximation)  
 ax.clear()  
 ax.plot\_surface(x, y, z, rstride=approximation, cstride=approximation, color='b', alpha=alpha, edgecolors="black")  
 max\_radius = max(rx, ry, rz)  
 for axis in 'xyz':  
 getattr(ax, 'set\_{}lim'.format(axis))((-max\_radius, max\_radius))  
 ax.grid(None)  
 ax.axis('off')  
 plt.draw()  
  
  
axbox = fig.add\_axes([0.184, 0.05, 0.31, 0.06])  
text\_box\_B = TextBox(axbox, "Approximation: ")  
text\_box\_B.on\_submit(change\_approximation)  
text\_box\_B.set\_val(str(steps - approximation))  
  
ax.grid(None)  
ax.axis('off')  
plt.show()

**Результат работы**

****





**Выводы**

Выполнив эту лабораторную работу, я научился рисовать трёхмерные фигуры с переходом в сферические координаты на компьютере при помощи библиотеки matplotlib. Также использовал интересные способы для аппроксимации и освещения.